Secure Software Distribution System

Tony Bartoletti (azb@llnl.gov),

Lauri A. Dobbs (dobbs1@llnl.gov),

Marcey Kelley (kelley6@llnl.gov)

Computer Security Technology Center

Lawrence Livermore National Laboratory

PO Box 808 L-303

Livermore, CA 94551

June 30, 1997

UCRL-JC-123354

Work performed under the auspices of the U. S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Secure Software Distribution System

Tony Bartoletti (azb@llnl.gov),
Lauri A. Dobbs (dobbs1@llnl.gov),
Marcey Kelley (kelley6@llnl.gov)

Computer Security Technology Center
Lawrence Livermore National Laboratory
PO Box 808 L-303
Livermore, CA 94551

June 30, 1997

Abstract

Authenticating and upgrading system software plays a critical role in information security, yet practical tools for assessing and installing software are lacking in today's marketplace. The Secure Software Distribution System (SSDS) will provide automated analysis, notification, distribution, and installation of security patches and related software to network-based computer systems in a vendor-independent fashion. SSDS will assist with the authentication of software by comparing the system's objects with the patch's objects. SSDS will monitor vendors' patch sites to determine when new patches are released and will upgrade system software on target systems automatically. This paper describes the design of SSDS. Motivations behind the project, the advantages of SSDS over existing tools as well as the current status of the project are also discussed.

Keywords: security, distributed, software management

Introduction

A serious threat to information resources is the inability to determine and maintain a known level of trust in operating system software. This threat can be minimized if systems are properly configured, use the latest software, and have the recommended security patches installed. However, the time and technique required to assess and install recommended security patches on systems is considerable and too often neglected. This situation is further complicated by the fact that each vendor has their own patch distribution and installation process. Some vendors are providing tools to assist with the installation process. Unfortunately, these "solutions" (self-installing patches, installation utilities) fail in three critical ways: 1) the target system is not actually examined; 2) the solutions are vendor specific; and 3) they operate on a single host as opposed to a multi-host networked solution.

The Secure Software Distribution System (SSDS) will provide automated analysis, notification, distribution, and installation of security patches and related software to network-based computer systems in a vendor-independent fashion. This system will allow a network administrator (and users) to query, maintain, and upgrade the software integrity of hundreds of individual systems from a central point through largely automated means. The centralized software system will provide the following services for target systems:

- Rapid system software "trust" determination.
- Automated notification of new vendor security patches.
- Automated determination of patch applicability to target systems.
- Automated installation of security patches and critical system software.
- Ability to "back-out" installed patches, restoring a system's previous state.
- Collection of site-wide software statistics or metrics on patch status.

The process SSDS will use to authenticate the software on a system is more reliable and secure than other vendor-specific tools. SSDS will compare the target system's objects with the objects from the patch to determine what is actually installed and what needs to be installed. This approach ensures accurate reporting of a system's patch status. It also allows SSDS to identify objects that do not belong to either the original system distribution or to any released patches.

Motivation

System software plays a central role in information security. Most technological methods for securing system resources are critically dependent upon the system software. Defining access control lists (ACLs), properly setting up user and group accounts, and configuration of network services is useless if the software that is supposed to be enforcing these parameters is not doing what is expected. Short of controlling the physical access to a system, assessing and maintaining the integrity of system software in a networked environment is the first step in information security.

System software is constantly changing, making it difficult to maintain its integrity. Often times, system software is security-flawed straight out of the box. Major network-wide assaults, such as the notorious 1988 Internet Worm attack, as well as a history of less publicized attacks, exploit these known security flaws to gain illicit access to systems. To their credit, vendors are often quick to issue security-patched versions of selected system files. Even when the vendors issue interim patches to fix the flaws, new releases of the system software may have overlooked these patches and/or introduced new flaws into the release. This is common in large software companies because the teams that create new releases are often different than the teams that create the interim software patches. This multiplicity of security patches and operating system versions significantly complicates the software authentication effort.

Even if system software was certifiably "clean," software authentication efforts must also be concerned with the possibility of tampering during episodes of weak security management. A common method of compromising the security of a workstation is to use a foothold on the system (*e.g.*, an unprotected user account) to modify key system files and compromise the system's defenses. Trojan horses are an example of this style of attack.

Vendors are aware of these issues and there is a push toward supplying the customers with "self-installing" patches or similar software installation packages to assist with the maintenance of software. However, these tools are highly vendor specific and vary wildly in their implementation and effectiveness. The tools we have encountered suffer from a common security flaw. These tools attempt to keep track of the patches that they have installed by building a "patch database" file. These tools can be easily fooled into reporting erroneous information because they make no attempt to survey the existing system files using secure cryptographic hashes or even ordinary checksums to ascertain what is actually there. For example, assume a patch to fix a vulnerability has been installed using such a tool. Subsequently, an intruder replaces the fixed binary with a Trojan or older flawed version. Using the tool to determine what is installed on the system will indicate that the patch is installed because the tool simply consults the "patch database" file. Solutions that rely on a database are unreliable and unacceptable for determining the level of "trust" in operating system software.

Additionally, existing tools do not address the problem of maintenance in a heterogeneous network environment. In an environment where a large mix of vendor systems are employed, the routine maintenance of software versions and interim patches is an administrative nightmare. Learning to operate and manage one vendor's set of software or patch management tools is grueling enough, let alone managing this task for all the flavors of UNIX and master their operations manuals. Including other popular operating systems such as Windows NT and MacOS complicates the maintenance task even more. Is there any wonder why installing patches and upgrades doesn't get done?

Software management tools are very much needed to support the assessment and authentication of system software on a network as well as installing and upgrading system software. Wouldn't it be nice to know exactly which of your 250 systems are patched up-to-date, which are not, and what patches are needed for each system? Sadly, there are many organizations where an administrator of 250 systems could not determine this in a month's time, and certainly not in a manner that involved the actual examination of installed binary files. SSDS will enable an administrator to produce this information within hours of the request, from a single console designed to support this type of inquiry and it will do so by actually examining the files present on these systems.

How much trust does a new network administrator place in the computer systems he/she just inherited? How much trust does an administrator place in a network recently experiencing suspicious activity? A responsible alternative to full network-wide system authentication would be to shut the machines down for a full re-install of their operating systems, along with the latest complement of security patches. This represents weeks of service disruption, and the assurances it gives will tend to dwindle with time. More often than not, this simply doesn't get done. Again, this is why a software management tool supporting software authentication is needed.

A software management tool should also support software re-authentication on a regular basis, commensurate in frequency with the value of the resources being maintained on the systems. SSDS will provide system administrators with a fast and highly automated method to authenticate system software, determine security patch versions and detect instances of subsequent tampering. In addition, it will provide a convenient and secure means for automating the installation of required security patches and related system software. Information security demands this capability at its foundation.

SSDS Architecture

A software management tool must be capable of collecting upgrades and patches; determining which upgrades and patches should be or have been applied to a system; and installing and possibly backing out upgrades and patches. Patches and upgrades can be collected from most vendors by downloading them from the vendors' ftp sites. To collect the newest releases, these ftp sites must be monitored regularly.

Once the patches are downloaded to the local system, a software management tool must determine which upgrades and patches should be or have been applied to a system. This is one of the most difficult tasks to automate in a software management tool. Each patch or upgrade must be interpreted to determine the operating system type, version and architecture the patch applies to; how much memory and disk space is needed to install the patch; dependencies on other layered products, patches, or upgrades; and which files and directories are affected by the installation of a patch. To determine which patches are installed on a system, the files on the system must be compared with files contained in each patch. At present this process is accomplished manually by reviewing the README file associated with each patch and upgrade.

If the patch is applicable to the system, then the software management tool can install the patch or upgrade. Installing a patch usually entails following a set of instructions provided with the patch or upgrade, or executing a script that will install the patch. Sometimes the patch doesn't work as advertised or it interferes with other applications on the system, so the software management tool must also permit patches or upgrades to be backed-out. Backing-out a patch or upgrade is similar to installation (*i.e.*, a set of instructions to follow or a script).

SSDS will largely automate the software management tasks described above. It will do this with two major software components: the SSDS Server and SSDS Agent installed on the target systems. Figure 1 illustrates these components along with their interaction with the vendors' ftp sites and other network-based computer systems.

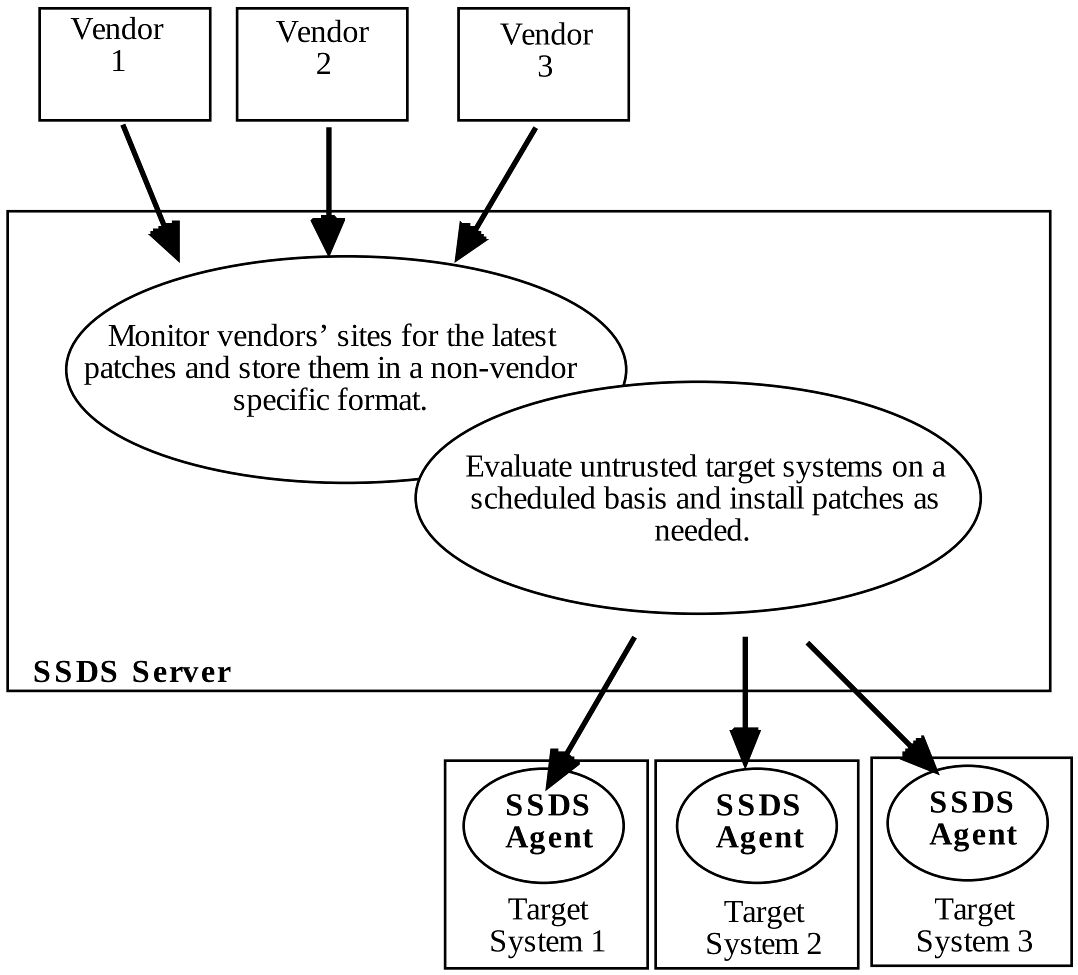
Vendor 1
Vendor 2
Vendor 3
Monitor vendors' sites for the latest patches and store them in a non-vendor specific format.
Evaluate untrusted target systems on a scheduled basis and install patches as needed.
SSDS Server
SSDS Agent
Target System 1
SSDS Agent
Target System 2
SSDS Agent
Target System 3

Figure 1

The SSDS Server software is a *central* service residing on one computer interacting with a host of network-based computer systems, similar to network-based backup software. The network-based computer systems serviced by the SSDS Server are referred to as **target systems** or **targets**. The SSDS Server is responsible for monitoring vendors' ftp sites and collecting newly released patches. The SSDS administrator can specify which vendor sites to monitor and which patches to collect (*e.g.*, security, recommended, all). For instance only Solaris 2.5+ security patches can be collected. These patches are then converted to a non-vendor specific, machine readable format and stored in a database. The process of converting patches will involve some human interaction until the vendors adopt a standard patch format. Patches stored in this format are referred to as patch specifications. A patch specification contains information such as the operating system type, version, and architecture as well as the access control list and ownership for each file and directory manipulated by the patch. A cryptographic checksum for each file is also included in the patch specification to be used for file identification during the evaluation process described later. A patch specification is built for every revision of each collected patch and stored in the patch spec database. By maintaining copies of all patches and all patch revisions, SSDS can determine what is installed on a system and if it is up-to-date. It is important for SSDS to collect every revision of a patch because the vendors’ ftp sites only provide the latest revision of all patches. Eventually we hope that the vendors will adopt a standard patch format or provide an adjunct for all of their patches (new as well as old patches).

In addition to collecting patches, the SSDS Server is responsible for evaluating target systems and installing patches on these systems. The system administrators have full control over the scheduling of target evaluations and patch installations. An administrator can request an evaluation immediately or can schedule evaluations on a repeated basis. The SSDS Server controls the execution of a request by gathering information from the target systems and giving it instructions to install and back-out a patch. To evaluate a system, the SSDS Server asks the SSDS Agent running on the target system what operating system, version, and architecture is running on the target. It then collects all of the patches from the patch spec database pertaining to this system’s operating system, version, and architecture. From these patch specifications a list of directories and files manipulated by the patch specifications is formed. The owner, access control list, and checksum (for files only) for each file or directory on the list is checked against the owner, access control list, and checksums of the respective directory or file on the target system. This check permits the SSDS Server to determine which patches are actually installed on the target system without relying on the system’s local database. From this information, the SSDS Server can determine which patches need to be installed on the target system in order to bring it up-to-date. The system administrator can choose to have SSDS install patches immediately after the evaluation or at some later date and time. The system administrator can also choose not to have SSDS install the patches and instead report on the patches needed. This allows for the system administrators to dictate which actions SSDS is to perform on a system.

The SSDS Agent software will reside on each target in order to respond to the SSDS Server's commands and requests. An SSDS Agent will be simple to install, easy to maintain, and use very few of the target's resources. This is why the majority of the work will be done by the centralized SSDS Server.

Secure communication between the SSDS Server and the SSDS Agent will be used to protect data from tampering and eaves-dropping and to authenticate services requested. Secure communications will employ digital signatures and encryption techniques based on public/private key technology. The SSDS Server will be implemented as a collection of independent processes. These processes will also communicate using digital signatures and encryption.

SSDS Use

SSDS can be configured to support most environments because each environment varies in the number and type of systems, as well as the system administration and security policies. SSDS will be easily configured to support both small homogenous networks and large heterogeneous networks.

For an organization with only a few computer systems (*e.g.*, 1 to 50 computers) all of the same type (*e.g.,* Macintosh, Sun) SSDS would be configured such that the SSDS Server resides on one computer and the SSDS Agent software would be installed on all of the computers including the computer running the SSDS Server software. As an example, consider a network of five Sun workstations running SunOS 4.1.4. One workstation would run the SSDS Server software. The SSDS Server would monitor Sun's ftp site for 4.1.4 patches and collect only these patches. The SSDS Agent software would reside on all five workstations. The SSDS Server would control the execution of the evaluations and installations based on the schedule determined by the SSDS administrator or each target's system administrator.

In a more complex environment with hundreds or thousands of systems running a variety of operating systems with different architectures there may be multiple SSDS Servers in order to service the large number of systems and networks. Lawrence Livermore National Laboratory (LLNL) is a good example of a complex environment. In a complex environment such as LLNL, the idea of centralizing the SSDS services is taken one step further. Here one or two computers would be configured to support the patch collection and storage function of the SSDS Server. This centralized patch collection service may be manned by one or two people to assist with the conversion of patches to the standard patch format until vendors adopt a standard format.

In addition to the one or two patch collection services, one system per subnet or organization would be configured to support the evaluation and installation of patches on a subset of LLNL's computer systems. The number of systems performing the evaluation and installation service would be determined by administrative domains similar to centralized services (*e.g.,* backups, mail). These systems would get their patches from the centralized patch collectors (see Figure 2). The SSDS Agent software would be installed on all systems in the network. This configuration distributes the work load and reduces duplication of effort.

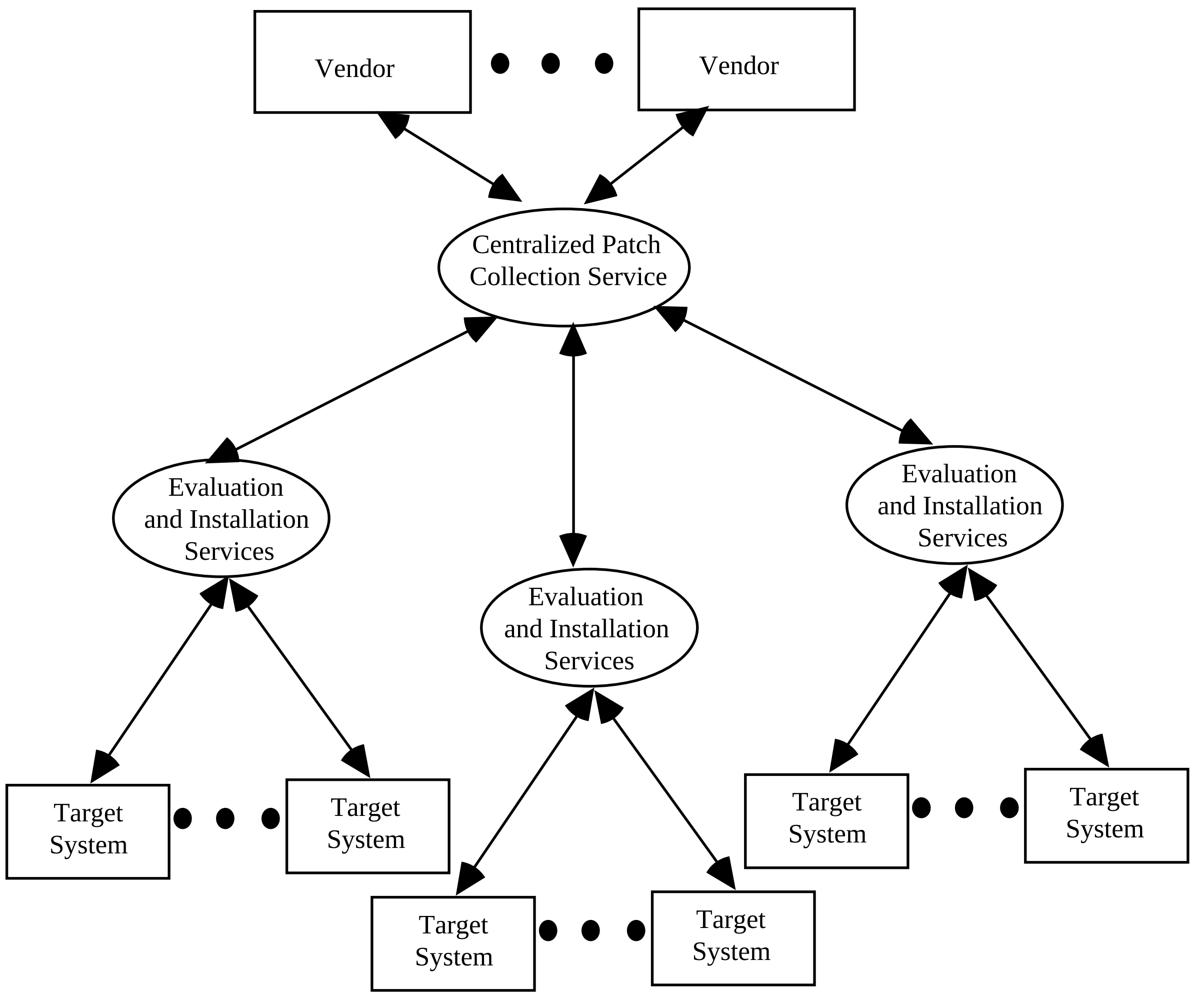
Vendor
Vendor
Centralized Patch Collection Service
Evaluation and Installation Services
Evaluation and Installation Services
Evaluation and Installation Services
Target System
Target System
Target System
Target System
Target System
Target System

Figure 2

Once SSDS is installed, the system administrator can schedule evaluations of one or more target systems on a periodic basis. For instance, all of the Sun Solaris systems can be evaluated and new patches installed every other Friday starting at midnight while the SGIs can be evaluated every other Saturday starting at nine in the morning. The scheduling and grouping of target systems will be flexible and up to the system administrator to determine. SSDS will also permit immediate queries of the target systems such as “is a particular patch installed?” or “does a particular object reside on one or more systems?” In the case where an intruder has been detected on a system and has left Trojan binaries, SSDS’s query capability can be used to detect these binaries. First construct a patch specification listing the Trojan binaries and their checksum. Then using SSDS, query all target systems to determine if this “patch” is installed on any of these systems. If this patch is detected on any of the systems, a full SSDS patch evaluation can be performed and the appropriate patches can be installed reverting the system back to the state before the intrusion.

Issues

SSDS provides system administrators with a tool to assist with the authentication of system software and automate the installation of security patches. However, the following issues relating to patches exist and are not currently addressed by the SSDS system.

In general patches fall into two categories: patches available for public access and patches available to those with service contracts or licenses. Security patches usually fall into the first category and are released via an ftp or web site. Currently SSDS is focused on installation of patches available without a service contract or license. In the future SSDS could be expanded to check licensing or service contract status and install a more complete set of patches for licensed systems or sites.

Another issue with vendors' patches is authentication of patches retrieved from the vendors' ftp sites (*i.e.*, patches have not been tampered with). This is an issue for anyone or any tool such as SSDS that retrieves patches from vendors' ftp sites. The COAST (Computer Operations, Audits, and Security Technology) Secure Patch Distribution Group at Purdue University is addressing this issue[1].

Finally, the integrity of vendors' patches is the biggest issue with automated patching systems. Often times a patch inconsistently reports the information necessary to determine whether the patch should be applied. For instance, one release of a patch indicates that it is a security patch, but future releases of the patch do not label it as a security patch. Due to this problem, SSDS could easily overlook the incorrectly labeled patches. Another frequent problem is with incorrect or incomplete installation instructions. Patches that conflict with other patches are another common problem. With SSDS, the patch specifications in the patch spec database are only as good as the patches retrieved from the vendors. These kinds of errors in patches can only be identified and corrected by a trained system administrator. In order to utilize SSDS to its fullest extent it should be coupled with a manual procedure where a trained system administrator would review each patch before submitting it to the patch spec database. The system administrator could also test each patch before submitting it to further increase the integrity of the patch spec database. Errors or incompatibilities with the patches would then be identified and corrected once instead of many times (*i.e.*, once by every system administrator in the whole organization).

SSDS Today and Tomorrow

SSDS development is funded by the Department of Energy's (DOE) Energy Research program. The project started in April of 1996. The goal of the project is to develop a proof-of-concept prototype over several phases of development. The first phase demonstrated the capability of detecting patch deficiencies on Sun systems running SunOS or Solaris. System deficiencies are listed in a detailed report. The report indicates which patches need to be installed, which order they should be installed, which patches

[1] See http://www.cs.purdue.edu/coast/projects/patch.html for more information about COAST's Security Patch Distribution.

are installed, and what binaries were unrecognized[2]. A graphical user interface is provided to support the scheduling, monitoring, and reporting of evaluation jobs. This phase is limited to patches that replace executables and/or manipulate directory and file permissions and ownership. Patches that manipulate shared libraries or require editing of configuration files will be addressed in a later phase of development.

The second phase of development will focus on the automated collection of patches from the vendors' patch sites as well as the conversion of these patches to a standard patch format. We have been successful in collecting patches from the vendors' patch sites and are working on developing tools to convert patches to a standard format. Conversion of some vendor patches should be relatively simple due to the development of some vendor specific tools such as Sun's install scripts. However, converting patches from other vendors will prove more difficult. The second phase of the project will also address the issue of secure communication between the networked processes.

The first release of SSDS will be in October 1997. This release will include the capability to detect deficiencies in Sun systems, the graphical user interface developed in the first phase, an automated patch collection service, patch conversion tools for Sun patches, and a Sun patch spec database containing Sun security patches. We will also port SSDS to other UNIX platforms including HP-UX and Digital UNIX. Minor changes to the software are expected for these ports.

Finally, the third phase of the project will focus on the automated installation and backing-out of patches. We will also extend the types of patches to include patches that manipulate shared libraries or require editing of configuration files. We will be starting this phase in October of 1997. Future extensions may support license-tracking to detect the presence of unlicensed software applications as well as the installation of other security software such as LLNL's Security Profile Inspector (SPI) and Network Intrusion Detector (NID).

Conclusion

Automated information processing is destined to play an increasingly important role in our lives, and it will become critically important to assure trust in these information systems. SSDS can fulfill a central role in this assurance with a uniform solution to the automated authentication and maintenance of system software. SSDS will serve to protect against threats to information resources and provide a high level of trust to the systems' users. We will continue our efforts in working with the vendors and encourage them to adopt a standard patch format. The future of the project is very promising and has received positive reviews.

[2] If the checksum of a binary or file does not match the checksum of any object belonging to the operating system's original distribution or to a patch then the binary is unrecognized. This binary may have been modified by the system administrator or worse it could be Trojan or malicious software placed on the system by some hacker.